PCT/IB04/52259

Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

03078437.5

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 03078437.5
Demande no:

Anmeldetag:
Date of filing: 03.11.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Koninklijke Philips Electronics N.V.
Groenewoudseweg 1
5621 BA Eindhoven
PAYS-BAS

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Simple and efficient multiparty computation based on homomorphic threshold ElGamal with applications

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F17/60

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

Simple and Efficient Multiparty Computation based on Homomorphic Threshold ElGamal with Applications

Abstract. We present new results in the framework of secure multiparty computation based on homomorphic threshold cryptosystems. We introduce the *conditional gate* as a special type of multiplication gate that can be realized in a surprisingly simple and efficient way using just standard homomorphic threshold ElGamal encryption. As addition gates are essentially for free, the conditional gate not only allows for building a circuit for any function, but actually yields efficient circuits for a wide range of tasks.

Our results are best viewed as uniting the benefits of the Mix and Match approach of Jakobsson and Juels [JJ00] and the approach of Cramer, Damgård, Nielsen [CDN01]. Like [CDN01], we take full advantage of the homomorphic properties of the underlying cryptosystem by not restricting the elementary gates to work on bits only, but on ring or field elements as much as possible. At the same time, like [JJ00], our results hold under the standard Decision Diffie-Hellman assumption using just homomorphic threshold ElGamal encryption, whereas the results of [CDN01] rely on the use of RSA-like cryptosystems such as Paillier's cryptosystem. This is also of great practical significance, as the generation of a shared RSA modulus for the corresponding threshold cryptosytems is costly, often dominating the cost of an entire application. Even for the two-party case, sharing an RSA modulus is a non-trivial task. In contrast, distributed key generation for discrete log based cryptosystems is simple, and practically for free in the two-party case.

We show that our approach leads to probably the most efficient solution to date for Yao's millionaires problem and many other problems, such as secure auctions, noting that we treat the malicious case and address fairness for the two-party case throughout. We observe that our approach performs particularly well for *ad hoc contacts* among a large group of peer users, where it is important that each user needs only a limited amount of set-up information (independent of the total number of users), and the total time of execution—including the time for distributed key generation—for running a protocol between any two users is limited as well. For example, we obtain practical protocols for problems such as "profile matching", where two parties jointly test whether some function of their profiles exceeds a given threshold without divulging any information on their profiles.

1 Introduction

The manual of any programming language will cover the arithmetic and relational operators in one of its opening chapters. Clearly, programming languages provide complete sets of operators that allow us to program any computable function. For convenience and for efficiency, though, the set of operators is larger than required from the computability point of view. Similarly, in secure multiparty computation one often considers addition

and multiplication (of bits) as the basic operations from which any function can be built. In this paper we set out to devise *efficient* building blocks for more advanced arithmetic and relational operators. These building blocks then allow us to quickly obtain practical solutions for many non-trivial tasks.

Homomorphic threshold cryptosystems provide a basis for secure multiparty computation. For a given n-ary function f, one composes a circuit of elementary gates that given encryptions of $x_1, \ldots, x_n$ on its input wires, produces an encryption of $f(x_1, \ldots, x_n)$ on its output wire. The elementary gates operate in the same fashion. The wires of the entire circuit are all encrypted under the same public key; the corresponding private key is shared among a group of parties. It is customary to distinguish addition gates and multiplication gates. Addition gates can be evaluated without having to decrypt any value, taking full advantage of the homomorphic property of the cryptosystem. Multiplication gates, however, require at least one threshold decryption to succeed even for the semi-honest case. To deal with the malicious case, multiplication gates additionally require the use of zero-knowledge proofs.

The elementary gates operate on bits (the field $\mathbb{F}_2$) or on elements of larger domains (rings or fields), where apparently the latter type is preferred from an efficiency point of view. Indeed [CDN01] (see also [DN03]) lays out a general approach to multiparty computation based on threshold homomorphic encryption, where all operations are defined for a (large) ring. This allows for very efficient arithmetic. Problems that require a lot of arithmetic may thus be solved efficiently. E.g., see [CD01] for recent results in this area.

For many problems however elementary gates operating on bits are sufficient. For example, all efficient solutions to Yao's millionaires problem rely on viewing the inputs as bit strings. In such cases the full generality of the protocols in [CDN01] is not needed, hence allowing for the use of a weaker form of homomorphic threshold cryptosystems. The homomorphic cryptosystems required by [CDN01] are known to exist only under RSA-like assumptions (e.g., Paillier's cryptosystem and its generalizations, and cryptosystems based on quadratic residuosity as used by Franklin and Haber in [FH96]). We show that basically homomorphic ElGamal suffices for efficiently handling a wide range of problems. The major advantage of such a discrete log based solution is that distributed key generation for the threshold version is relatively simple (see [Ped91,GJKR99]), while generating a shared RSA modulus, even when just two parties are involved, is far from trivial. (For two parties: RSA best result: $O(k^4)$ (Crypto 2002, p.431, honest-but-curious) whereas DL: $O(k^3)$) bit complexity; general $O(nt(k + t \log n))$ modular multiplications). As an additional advantage, ElGamal allows for solutions based on any discrete log setting, such as elliptic curves or XTR.

Multiparty computations based on homomorphic threshold cryptosystems typically have a low computational complexity and a low bit complexity, but a high round complexity. For applications such as integer comparison (Yao's millionaires problem) our protocols require considerably less work and communication than [JJ00,CDN01]. The price we pay is that the round complexity for our protocols may be higher. However, for the standard millionaires problem the round complexity of our solution is the same as [JJ00,CDN01], and likewise for many other cases the round complexity is low.

We observe that for many applications our approach performs particularly well for *ad hoc contacts* among a large group of peer users, where it is important that each user needs only a limited amount of set-up information (independent of the total number of users), and the total time of execution—including the time for distributed key generation—for running a protocol between any two users is limited as well. For example, we obtain practical protocols for problems such as "profile matching", where two parties jointly test whether some function of their profiles exceeds a given threshold without divulging any information on their profiles.

2 Introduction

Homomorphic encryption schemes are very attractive as they allow to make computations with encrypted values. Some well known examples of these kind of computations are auctions [Bau01], the Millionaires problem [Yao82], secure function evaluation [JJ00], voting [?], crypto computing with rationals [FSW01] and secure profile matching etc.

A basic tool in the toolbox for computing under the encryption, is a secure multiplication protocol. In [CDN01] a general multiparty multiplication protocol was developped. However in its full generality the protocols in [CDN01] rely on threshold homomorphic encryption systems that satisfy a rather strong property: the decryption protocol is required to return the exact plaintext. Such cryptosystems follow from Paillier's cryptosystem and its generalisations or from cryptosystems based on quadratic residuosity (or higher order residues). These systems all use an RSA modulus. There are several disadvantages to this. First, no discrete log alternatives are known. Secondly, obtaining threshold versions of these schemes are notoriously hard as generating a shared RSA modulus is complicated.

3 The Problem

A two-party multiplication protocol is a protocol carried out by two players. The input of the protocol consists of two (possibly encrypted) numbers, say x and y. The number x is provided by player 1 and y is provided by player 2. At the end of the protocol, both players get the product $[\![xy]\!]$ as a result. Moreover they get a proof that the result was correctly computed and that the other player did not cheat. More specifically, we focus on the following situation: Both players have only an encrypted version of their input, $[\![x]\!]$ and $[\![y]\!]$ respectively.

We will refer to this protocol as a multiplication protocol with a shared dichotomous multiplier. The protocols that have been developped in the literature [CDN01] are very general but not so efficient.

4 The solution

The focus of the solution of this paper is on *efficient* multiplication protocols. General secure multi-party multiplication protocols were developped in [CDN01]. By considering some concrete practical situations, more efficient solutions can be obtained.

The scheme that we propose is based on the ELGamal cryptosystem explained elsewhere in this document.

4.1 Multiplication protocols

A multiplication protocol takes as input two encrypted values x and y, say, and produces an encryption of the product xy. Instead of encryptions, also commitments can be used. We like to use weak-homomorphic encryption schemes only. As noted above, such ElGamal-like homomorphic schemes do not allow for a general multiplication protocol. Therefore, we consider a special case for multiplication.

If both inputs are shared, then we cannot have an efficient multiplication protocol in general. However, if we assume that one of the shared inputs is two-valued, an efficient multiplication protocol can still be constructed.

5 The problem

In this we provide an answer for the following problem. Two users, Alice and Bob, have a profile which expresses their interests in a specific item. As an example one might think of Alice and Bob having a database of music files. They add to each song a number expressing how much they like it. When they meet each other, they would like to compare their profiles without exchanging the details of their profiles. In a more abstract way, we solve the problem of allowing two users to compare their private data without revealing any other information than whether they are similar or not (according to some predetermined measure).

As a spin-off of this solution, we describe also a few protocols for searching in encrypted databases (membership tests) and auctions.

6 Homomorphic Commitments and Encryptions

6.1 Preliminaries

Discrete Log Setting. Let $G = \langle g \rangle$ denote a finite cyclic (multiplicative) group of prime order q for which the Decision Diffie-Hellman (DDH) problem is assumed to be infeasible: given $g^\alpha, g^\beta, g^\gamma \in_R G$, it is infeasible to decide whether $\alpha\beta \equiv \gamma \pmod q$. This implies that the discrete log (DL) problem is infeasible as well: given $h = g^\alpha \in_R G$, it is infeasible to compute $\log_g h = \alpha$.

Σ-Protocols. We briefly mention a few facts about Σ-protocols [CDS94]. A Σ-protocol is a three-move proof of knowledge satisfying special soundness and special honest-verifier zeroknowledge. In [CDN01] a method is given to generate the challenge for the Σ-protocols jointly. We will employ the random oracle model.

Homomorphic ElGamal Encryption. For public key $h \in G$, we use additively homomorphic ElGamal encryption, where message $m \in \mathbb{Z}_q$ is encrypted as a pair $(a, b) = (g^r, g^m h^r)$, with $r \in_R \mathbb{Z}_q$. The homomorphic property is that componentwise multiplication of encryptions of m and m', respectively, yields an encryption of $m + m'$ (modulo q):

$$(a, b) * (a', b') = (aa', bb') = (g^{r+r'}, g^{m+m'} h^{r+r'}).$$

Given an encryption $(a, b) = (g^r, g^m h^r)$ as common input, standard techniques yield a proof of knowledge for showing knowledge of the (unique) witness (m, r). (Standard ElGamal encryption with encryptions of the form (g^r, mh^r) for $m \in G$ is homomorphic in a multiplicative sense but lacks such a proof of knowledge.)

We define an equivalence relation on $G \times G$ by stating that encryptions (a, b) and (a', b') are equivalent iff $\log_g(a/a') = \log_h(b/b')$. Using $(1, g^m)$, $m \in \mathbb{Z}_q$, as canonical representatives, we use $[\![m]\!]$ to denote the equivalence class of $(1, g^m)$. In other words, $[\![m]\!]$ denotes the set of all ElGamal encryptions of m (under public key h). The operations on the direct product group $G \times G$ are lifted to the equivalence classes in the usual way. The homomorphic property then implies that:

$$[\![x]\!] * [\![y]\!] = [\![x + y]\!].$$

$$[\![x]\!]^c = [\![cx]\!].$$

Thus addition and multiplication by a scalar are easily accomplished. These operations can easily be verified when we implement these in a deterministic fashion.

Randomization (or blinding) of ElGamal encryptions is an important primitive as well. This amounts to multiplying a given encryption with a random element $(a, b) \in_R [\![0]\!]$. Proving that $\log_g a = \log_h b$ shows that (a, b) is indeed an encryption of 0.

Given the private key $\alpha = \log_g h$, decryption is performed by calculating b/a^α, which is equal to g^m for some $m \in \mathbb{Z}_q$. Recovering m from g^m is supposed to be hard in general, hence we need to view this cryptosystem with respect to a set $M \in \mathbb{Z}_q$ of sufficiently small size such that finding m from g^m is feasible whenever $m \in M$. In this paper, however, the size of M will be very small, often $|M| = 2$.

The ElGamal cryptosystem is semantically secure under the DDH assumption, but clearly it lacks security against active attacks due to the homomorphic properties.

Pedersen Commitment. Given $g, h \in G$, a commitment to message $m \in \mathbb{Z}_q$ is a value $c = g^m h^r$, with $r \in_R \mathbb{Z}_q$. The commitment is opened by revealing m and r. Pedersen's scheme is unconditionally hiding and computationally binding, under the assumption that $\log_g h$ cannot be determined. The commitment scheme is also additively homomorphic, and we will use $(\!(m)\!)$ to denote a commitment to message m, where the randomization is suppressed.

Threshold ElGamal Decryption. In a (t, n)-threshold version of ElGamal, $1 \leq t \leq n$, encryptions are computed w.r.t. a common public key h (as above) while decryptions are done using a joint protocol between n parties, each party possessing a share of the private

key $\alpha = \log_g h$. As long as at least t parties take part, decryption will succeed, whereas fewer than t parties are not able to decrypt successfully. The parties obtain their share by running a distributed key generation protocol. See [Ped91,GJKR99] for details.

Since we are particularly interested in two-party computation we give some more details for the $(2,2)$-threshold scheme. Distributed key generation is achieved by having parties P_1, P_2 first broadcast commitments $c_i = g^{\alpha_i} h^{r_i}$, with $\alpha_i, r_i \in_R \mathbb{Z}_q$ for $i = 1, 2$, and then broadcast the values r_i along with proofs of knowledge of $\log_g h_i$, where $h_i = c_i / h^{r_i}$ for $i = 1, 2$. The joint public key is $h = h_1 h_2$, with private key $\alpha = \alpha_1 + \alpha_2$. To decrypt an encryption (a, b), player P_i produces $d_i = a^{\alpha_i}$, along with a proof that $\log_a d_i$ is equal to $\log_g h_i$. The message is then recovered from $b/(a_1 a_2)$.

Clearly, $(2,2)$-threshold ElGamal allows for ad-hoc use. The effort for generating the keys is about the same as the effort for performing a decryption.

7 Multiplication protocols

In designing efficient circuits we take as much advantage as possible form the fact that the message space $\mathbb{Z}_q$ is actually a field. Due to the homomorphic properties of ElGamal encryption, addition over $\mathbb{Z}_q$ is essentially for free. The same applies to multiplication with a publicly known constant in $\mathbb{Z}_q$. In this section, we are concerned with multiplications where both inputs are encrypted.

A multiplication protocol takes as input an encrypted multiplier x and an encrypted multiplicand y and produces in polynomial time as output an encryption of the product xy. The protocol should not leak any information on x, y, and xy. Furthermore, it is required that the protocol generates a publicly verifiable proof that the product is computed directly. Thus, the security of the protocols is considered for the malicious case.

The encryptions of x, y, and xy are homomorphic ElGamal encryptions denoted by $[\![x]\!]$, $[\![y]\!]$, and $[\![xy]\!]$, where it is understood that these encryptions are randomized and the public key for these encryptions is always the same. The corresponding private key is shared among a number of parties.

If no restrictions are put on x or y, such a multiplication protocol cannot exist under the Diffie-Hellman assumption. (Given g^x, g^y we form homomorphic ElGamal encryptions for x and y respectively. The multiplication protocol would return a homomorphic ElGamal encryption of xy, which would give g^{xy} upon decryption.) Therefore, we consider two special multiplication protocols, putting some restrictions on the multiplier x. The first protocol, referred to as the *conditional gate*, requires that the multiplier x is from a *dichotomous* (two-valued) domain. The second protocol requires that the multiplier x is private, that is, known by a single party.

Finally, we also consider

7.1 Multiplication with a Private Multiplier

In this section, we present a multiplication protocol where the multiplier x is a *private* input rather than a shared input. That is, the value of x is known by a single party P. No

restriction is put on the multiplicand y. Multiplication with a private multiplier occurs as a subprotocol in the protocol for a conditional gate and other protocols further on in the paper.

Given encryptions $[\![x]\!], [\![y]\!]$, where party P knows α, x such that $[\![x]\!] = (g^\alpha, h^\alpha g^x)$, party P computes an encryption $[\![xy]\!]$ on its own, along with a publicly verifiable proof showing that the output is correct. Due to the homomorphic properties, party P can do so in a straightforward way—without the need for decryptions.

Let $[\![x]\!] = (A, B) = (g^\alpha, h^\alpha g^x)$, where party P knows α, x, and let $[\![y]\!] = (C, D)$. Player P broadcasts $[\![xy]\!] = (E, F) := (g^\gamma C^x, h^\gamma D^x)$, along with a non-interactive proof that it knows α, x, γ such that

$$A = g^\alpha, \quad B = h^\alpha g^x, \quad E = g^\gamma C^x, \quad F = h^\gamma D^x.$$

This is a proof that requires 4 (multi-)exponentiations for both the prover and the verifier; it consists of 3 values from $\mathbb{Z}_q$.

A simple modification to the above protocol lets party P use a commitment $(\!(x)\!)$ instead of $[\![x]\!]$, where $(\!(x)\!) = f^\alpha g^x$. Another variation is to multiply x with several multiplicands y_i at the same time. In such cases, the non-interactive proofs can be done more efficiently.

7.2 Conditional Gates: Multiplication with a Shared Dichotomous Multiplier

Consider a multiplication operation xy where the multiplier x is from a dichotomous (two-valued) domain, whereas the multiplicand y is from an unrestricted domain. In this section we show that dichotomous multipliers allow for a simple multiplication protocol, to which we refer as a *conditional gate*. As will be shown later in this paper, these conditional gates form a very powerful primitive.

The dichotomous domain $\{-1, 1\}$ is convenient for our purposes. Domain $\{0, 1\}$ or any other domain $\{a, b\}$, $a \neq b$, can be used instead, as these domains can be transformed into each other by linear transformations: $x \mapsto a' + (b' - a')(x - a)/(b - a)$ maps $\{a, b\}$ onto $\{a', b'\}$. These transformations can be applied to encryptions, transforming $[\![x]\!]$ with $x \in \{a, b\}$ into $[\![x']\!]$ with $x' \in \{a', b'\}$.

Let $[\![x]\!], [\![y]\!]$ denote encryptions, where it is given that $x \in \{-1, 1\}$. The following protocol enables players $P_1, \ldots, P_N$, $N \geq 2$, to compute an encryption $[\![xy]\!]$ securely. For simplicity, we assume that the players $P_1, \ldots, P_N$ also share the private key of the homomorphic encryption scheme $[\![\cdot]\!]$.

Stage 1 For $i = 1, \ldots, N$, player P_i takes $[\![x_{i-1}]\!]$ as input and chooses $s_i \in_R \{-1, 1\}$. Player P_i broadcasts encryptions $[\![s_i]\!]$ and $[\![s_i x_{i-1}]\!]$, and a proof that $[\![s_i x_{i-1}]\!]$ is correct w.r.t. $[\![s_i]\!]$ and $[\![x_{i-1}]\!]$, using the protocol for multiplication with a private multiplier. Let $x_i = s_i x_{i-1}$.

Stage 2 The players jointly decrypt $[\![x_N]\!]$ to obtain x_N. Each player checks that $x_N \in \{-1, 1\}$. Given x_N and $[\![y]\!]$, the encryption $[\![x_N y]\!]$ is computed publicly. Let $z_0 = x_N y$.

Stage 3 For $i = 1, \ldots, N$, player P_i takes $[\![z_{i-1}]\!]$ as input and broadcasts an encryption $[\![s_i z_{i-1}]\!]$, and a proof that $[\![s_i z_{i-1}]\!]$ is correct w.r.t. $[\![s_i]\!]$ and $[\![z_{i-1}]\!]$, using the protocol for multiplication with a private multiplier. Let $z_i = s_i z_{i-1}$.

The output of the protocol is $[\![z_N]\!] = [\![xy]\!]$. Note that the protocol requires a single threshold decryption only. Since $x_N \in_R \{-1, 1\}$ must hold, decryption is feasible for the homomorphic ElGamal encryption scheme. As the value of x_N is statistically independent if at least $t = N/2$ honest players are able to complete the protocol successfully, the value of x_N does not reveal any information on x.

The protocol requires roughly $2N$ rounds.

Optimistic mode: run as above.

The protocol can be made robust as follows. If a player P_i fails in stage 7.2, it is simply discarded in the remainder of the protocol. For stage 7.2, the joint decryption step is robust by definition. If the check $x_N \in \{-1, 1\}$ fails, the players are required to broadcast a proof that $s_i \in \{-1, 1\}$. The players who fail to provide a correct proof are discarded, and their s_i values are decrypted. The value of x_N is adjusted accordingly. Similarly, in stage 7.2, if player P_i fails to complete its step, its value s_i is decrypted and the encryption $[\![s_i z_{i-1}]\!]$ is computed publicly.

Theorem 1. *The protocol is correct, sound, and computational zk.*

Proof. Clearly, $z_N = xy(\prod_{i=1}^N s_i)^2 = xy$ is the desired output if the parties are honest.

The soundness of the proofs in step 7.2 ensure that $x_N = x \prod_{i=1}^N s_i$. Since it is checked in step 7.2 that $x_N \in \{-1, 1\}$, it follows from $x \in \{-1, 1\}$ that $\prod_{i=1}^N s_i \in \{-1, 1\}$ as well. The soundness of the proofs in step 7.2 ensure that $z_N = x_N y(\prod_{i=1}^N s_i)$, hence that $z_N = xy$.

For zero-knowledgeness, we may simulate all but one player w.l.o.g. P_1 as follows. The simulator extracts the values s_i for $i \neq 1$ and simulates P_1 by setting $s_1 = \widetilde{s_1}/x$ (using 'compatibility' of $[\![\cdot]\!]$ and $(\!(\cdot)\!)$ ...) and simulating the proof in step 7.2. Then the simulator is able to produce the decryption in step 7.2 as $\widetilde{s_1} s_2 \cdots s_N$. To perform step 7.2, compute $[\![s_1 z_0]\!]$ as $[\![\widetilde{s_1} z_0]\!][\![x]\!]$ and simulate the proof (these encryptions are computationally indistinguishable).

Intuition: only a random ciphertext is decrypted

NOTE: handle threshold decryption.

Note that we do not need that each s_i is in $\{-1, 1\}$. We assume that at least one player is honest, say P_1. If $N > 2$, however, two of the remaining players, say P_2 and P_3, may set $s_2 = 2$ and $s_3 = 1/2$, such that $s_2 s_3 = 1$. This is harmless to the protocol.

8 Logical gates

Any operator on two bits x and y can be expressed in a *unique* way as a polynomial of the form:

$$a_0 + a_1 x + a_2 y + a_3 xy.$$

The coefficients are not necessarily binary. For example, the exclusive-or operator $\oplus$ satisfies $x \oplus y = x + y - 2xy$. There are exactly 16 polynomials of type $\{0, 1\}^2 \to \{0, 1\}$, which is immediate if one considers the following normal form:

$$b_0 xy + b_1 x(1 - y) + b_2(1 - x)y + b_3(1 - x)(1 - y),$$

where the coefficients are binary. The correspondence is given by:

$$\begin{aligned} a_0 &= b_3, \\ a_1 &= b_1 - b_3, \\ a_2 &= b_2 - b_3, \\ a_3 &= b_0 - b_1 - b_2 + b_3. \end{aligned}$$

In general, the coefficients need not be integers either, if one works with other two-valued domains such as $\{-1, 1\}$.

8.1 XOR-Homomorphic Encryption from Homomorphic ElGamal Encryption

The well-known scheme of Goldwasser and Micali based on the Quadratic Residuosity assumption [GM84] is a basic example of an xor-homomorphic encryption scheme, that is, the scheme is homomorphic w.r.t. $(\mathbb{Z}_2, +)$. A natural question is how xor-homomorphic and $(\mathbb{Z}_n, +)$-homomorphic schemes with $n > 2$ relate to each other. In [KMO01], the problem of building $(\mathbb{Z}_n, +)$-homomorphic schemes from xor-homomorphic schemes is considered; the motivation being that not all encryption schemes are $(\mathbb{Z}_n, +)$-homomorphic. However, it is also true that not all encryption schemes are xor-homomorphic. In particular, it is hard to obtain a xor-homomorphic ElGamal scheme.

As a direct application of the conditional gate, we obtain an xor-homomorphic ElGamal encryption scheme. Given $[\![x]\!]$ and $[\![y]\!]$ with $x, y \in \{0, 1\}$, we compute $[\![x \oplus y]\!]$ as follows:

1. Publicly convert $[\![x]\!]$ to $[\![x']\!]$ with $x' = 2x - 1 \in \{-1, 1\}$.
2. Apply the conditional gate to $[\![x']\!]$ and $[\![y]\!]$ to obtain $[\![x'y]\!]$.
3. Publicly compute $[\![x - y - x'y]\!]$, which is equal to $[\![x \oplus y]\!]$.

The application of the conditional gate requires a threshold decryption. However, this seems unavoidable for achieving xor-homomorphic ElGamal encryption.

8.2 Fairness

Recall that t denotes the maximum number of corrupted parties tolerated by the circuit evaluation protocol. So far we have focused on the case that $t < n/2$, that is, the case of a dishonest minority, for which the protocol achieves robustness. We now extend the protocol to handle the two-party case $t = n/2 = 1$.

For the two-party case we give up on robustness, as it is well-known that robustness cannot be achieved for dishonest majorities. In other words, one cannot prevent one of the parties from quitting the protocol prematurely. If a party chooses to do so, however, it should not gain any advantage from it. If a protocol achieves this property, the protocol is said to be *fair*.

Now, the important observation for the above circuit evaluation protocol is that neither party gains any advantage from quitting the protocol in phase 1 or phase 2 of the

protocol. In particular, consider the case that party P_2, say, chooses to quit during the threshold decryption step of a conditional gate, for which party P_1 already produced its decryption share. In that case, only P_2 learns the decrypted value, but this value cannot possibly give P_2 an advantage, as this value is statistically independent of the inputs to the conditional gate. This line of reasoning also applies to the case that many conditional gates are evaluated in parallel.

To achieve fairness, we only need to protect the decryption of the output values. For this purpose, we will apply a protocol somewhat similar to that of [BST01]. In [BST01], however, the protocol steps for achieving fairness are intertwined with the original protocol steps, while in our protocol the additional steps for achieving fairness are strictly limited to the decryption of the output values.

Let encryption (a, b) be given. Recall from Section *hencs* that $(2,2)$-threshold decryption, requires party P_i to provide a^{α_i}, $i = 1, 2$. Instead of directly revealing this value, we will release it gradually using the following protocol, where k is a security parameter, $k < \log_2 q$:

1. For $i = 1, 2$, party P_i chooses $\epsilon_{ij} \in_R \{0,1\}, \alpha_{ij} \in_R \mathbb{Z}_q$ for $j = 0, \ldots, k-1$ subject to the condition that $\alpha_i = \sum_{j=0}^{k-1} \alpha_{ij} 2^j$. Party P_i then broadcasts the values $c_{ij} = a^{\alpha_{ij}} g^{\epsilon_{ij}}$, $j = 0, \ldots, k-1$ along with a proof that each $\epsilon_{ij} \in \{0,1\}$ (see Section *hencs*).
2. Set $j = k-1$. Parties P_1, P_2 repeatedly execute the following step. For $i = 1, 2$, party P_i broadcasts values $\alpha_{ij}, \epsilon_{ij}$. If all these values verify correctly against c_{ij}, the value of j is decremented and the step is repeated if $j > 0$.
3. Once $j = 0$ both parties release ϵ_{i0} along with a proof of knowledge for a witness α_{i0} satisfying $c_{i0} g^{-\epsilon_{i0}} = a^{\alpha_{i0}}$.
4. Both parties are able to recover the missing value a^{α_i}, as follows:

$$a^{\alpha_i} = c_{i0} g^{-\epsilon_{i0}} a^{\sum_{j=1}^{k-1} \alpha_{ij} 2^j}.$$

At each stage of the protocol, either party is at most one bit ahead of the other party, hence the protocol is fair. If one sets $k = 80$, for instance, it is clearly infeasible for both parties to compute the missing value a^{α_i} at step 1, as it requires a search over 2^k possible values for $\epsilon_{i,k-1}, \ldots, \epsilon_{i0}$. At each later step, the search space is reduced in size by a factor of two.

The protocol does not leak any information on α_i beyond what is implied by the output values a^{α_i}. The protocol can be run in parallel for decrypting multiple outputs at the same time, and the protocol can be combined easily with our protocol for private outputs presented above.

The problem of achieving fairness for two-party computation was recently addressed in [?], for protocols based on Yao's "garbled circuit" approach. Our results show that fairness can be achieved in a simple and elegant way for any two-party computation for protocols based on homomorphic threshold cryptosystems.

9 General Circuits

9.1 Overview

In our approach a circuit for evaluating an arbitrary function will be build from addition gates and the special multiplication gates. Clearly, this set of gates is sufficient to build a circuit for any function. We will show though that our set of elementary gates allows for particularly efficient circuits for a wide range of problems.

The inputs elementary gates can be any value in $\mathbb{Z}_q$, except that the multiplier for the conditional gate must be binary.

1. Players provide encrypted inputs.
2. Players evaluate each gate of the circuit, where the outputs are put in encryptions.
3. Finally, decrypt the outputs.

Perform rounds at same depth in parallel.

9.2 Adding Fairness to Threshold Decryption

Gradually reveal decryption of final ciphertext. For n−party computations with $n > 2$, if one works with an honest majority, one achieved fairness automatically.

Consider the two-party case. Given an encryption $[\![x]\!]$, we wish to decrypt it in a fair way. Assume that $(2,2)$-threshold decryption is used. Instead of providing a^{x_1}, a player will provide $a^{x_1}g^e$, where $e \in_R \{0,\ldots,2^k-1\}$ for some security parameter k. In addition, the commitments $a^{x_1}g^{e_i}$ are provided plus 0/1 proofs etc. Then they may gradually release the e_i's.

If the output of the protocol consists of multiple encryptions, the gradual decryption is run in parallel on each decryption.

Note that the intermediate decryptions do not give any advantage to either player, when one of them chooses to abort the protocol. Therefore, only the final decryption needs to made fair using well-known standart techniques.

9.3 Private Outputs

Private outputs can be obtained in various ways. A well-known method lets the receiver blind the ciphertext (more precisely, blinding the plaintext contained in it; the receiver needs to prove that the blinding is done honestly, by showing that it knows the random plaintext used as one-time pad: prove knowledge of α, β in $(g^\alpha, h^\alpha g^\beta)$). The blinded ciphertext is decrypted, and only the receiver is able to unblind the plaintext. Note, however, An alternative method avoids interaction with the receiver: each share of the decryption is ElGamal encrypted by the players in the threshold scheme with the public key of the receiver, and a proof is given that the encrypted share is correct. The receiver only needs to decrypt the Lagrange interpolation of these encrypted shares.

10 Relational and Arithmetic Operators

In this section we will apply the special multiplication gates to obtain efficient circuits for basic operations such as integer comparison and addition of binary represented numbers.

10.1 The Millionaires

In this section, we present an effcient solution for a slight variant of Yao's millionaires problem that allows extensions to more general situations. We assume that $\mathbf{x}$ and $\mathbf{y}$ are given by their binary representations, i.e. $\mathbf{x} = (x_{n-1}, \ldots, x_0)$ and $\mathbf{y} = (y_{n-1}, \ldots, y_0)$ respectively. Firstly, we define a multivariate polynomial P over $\mathbb{Z}$ that implements the sign function, i.e. $P(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}) = \mathrm{sgn}(x - y)$ for all $x, y, 0 \leq x, y, \leq 2^n$. Hereby sgn is defined as:

$$\mathrm{sgn}(z) = \begin{cases} -1, z < 0, \\ 0, z = 0, \\ 1, z > 0. \end{cases} \tag{1}$$

Several polynomials can be used to implement this function. It turns out that the most efficient solution can be constructed based on the following multivariate reduction polynomial. For $x, y \in \{0, 1\}$, we define

$$F(s, x, y) = s + (1 - s^2)(x - y). \tag{2}$$

Note that $s^3 = s$, since $s \in \{-1, 0, 1\}$. Furthermore, it readily follows that $F(0, x, y) = \mathrm{sgn}(x - y)$ and that $F(1, x, y) = 1$. For $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ the polynomial P that implements the sgn-function is then constructed as follows,

$$P(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}) = F\left(\ldots F\left(F\left(0, x_{n-1}, y_{n-1}\right), x_{n-2}, y_{n-2}\right), \ldots, x_0, y_0\right) \tag{3}$$

It can be easily verified that the polynomial $P(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ computes $\mathrm{sgn}(\sum_{i=0}^{n-1}(x_i - y_i)2^i)$. The polynomial F can be efficiently evaluated by introducing an auxiliary variable $v = 1 - s^2$. Initially, we put $s = 0$ and $v = 1$. Then, we compute:

$$\begin{aligned} s &\leftarrow s + v(x - y); \\ v &\leftarrow v - v(x - y)^2. \end{aligned}$$

The repetition of this computation for all components of $\mathbf{x}$ and $\mathbf{y}$ gives the desired result. Note that $v - v(x - y)^2$ can be computed as $v(1 - x + 2xy - y)$. In order to do this computation in a private way, one needs three basic steps, where a player multiplies its x or y with a given homomorphic encryption.

1. Player 1 computes $[\![vx]\!]$ from $[\![v]\!]$ and $((x))$.
2. Player 2 computes $[\![vy]\!]$ and $[\![vxy]\!]$ from $[\![v]\!]$ resp. $[\![vx]\!]$ and $((y))$.
3. (Both may) Compute $[\![s + vx - vy]\!]$ (which is the new s).
4. (Both may) Compute $[\![v - vx + 2vxy - vy]\!]$ (which is the new v).

if needed, s can be decrypted using threshold decryption. Note that this algorithm needs three "multiplication with a private multiplier" protocols for each bit. Clearly, this algorithm has the following invariants:

$$s = \mathrm{sgn}(\sum_{i=j}^{n-1}(x_i - y_i)2^{i-j}) \quad v = \prod_{i=j}^{n-1}(1-(x_i-y_i)^2), \quad 0 \leq j \leq n.$$

The second step in the algorithm can be performed efficiently by the algorithm described in section ??. This approach can also be applied to the Socialist Millionaires problem to produce the result in encrypted form. Hence, it extends the solution described in [BST01]. The algorithm is based on a multivariate polynomial $Q(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ that implements the function $\delta(z)$ which is defined as $\delta(0) = 0$ and $\delta(z) = 1$ for $z \neq 0$. The reduction polynomial G used to construct Q is given by,

$$G(s, x, y) = (1-(x-y)^2)s + (x-y)^2,$$

where $x, y, s \in \{0, 1\}$. The polynomial Q is then given by,

$$Q(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}) = G\left(G \ldots (G\left(G\left(0, x_0, y_0\right), x_1, y_1\right), \ldots, x_{n-1}, y_{n-1}\right). \quad (4)$$

Clearly, an algorithm analogous and with the same complexity as the one mentioned above can be used to implement this function efficiently (with initial conditions $s = 0$, $j = 0$).

For the sake of completeness we present another solution of similar complexity as [BST01]. The players commit to $[\![x]\!]$ and $[\![y]\!]$, respectively. The two players jointly form $[\![r]\!]$, where r is uniformly distributed and neither of the players knows r. Using multiplication with a private multiplier, the players compute $[\![(x-y)r]\!]$ from $[\![x]\!]$, $[\![y]\!]$ and $[\![r]\!]$, to obtain $g^{(x-y)r}$ after decryption. If $g^{(x-y)r} = 1$, then w.v.h.p. $x = y$, otherwise $x \neq y$. In fact, this is very close to [BST01], which uses $(2, 2)$-threshold decryption in disguise.

10.2 Generalized Millionaires problems

In the millionaires problem, the respective inputs x and y are both private to the players. In many applications (e.g. secure profile matching), however, one or both of the inputs will be shared. We show how to extend our approach from the previous section 10.1 to this case.

From the considerations in section 10.1 it follows that if only one input is shared, say x, we can still use the multiplication with a private multiplier protocol at a few steps in the algorithms. For the millionaires algorithm this leads to $2n$ private multiplier protocols and n dichotomous multiplication protocols. If both inputs are shared however, we have to use the dichotomous multiplication protocol at all steps, giving $3n$ uses of the dichotomous multiplication protocols.

If one input is shared say x, and the other input is a known constant T, we can do the following. Replace y_j with T_j for $j = 0, \ldots, n-1$ and compute $\{[\![x_j - T_j]\!]\}_{j=0}^{n-1}$ by using the homomorphic properties of the encryption scheme. In this way the problem is transformed into the inequality $x - T > 0$. Then, only the computation of $[\![v(x_j - T_j)]\!]$ in the algorithm of section 10.1 has to be done with the dichotomous multiplication protocol (leading to n dichotomous multiplications).

10.3 Arithmetic Operators

To add two numbers x,y given by their binary representation, the respective bits are added, also taking the carry into account. To produce the next bit of the output z, we need to compute $[\![t]\!] = [\![x_i + y_i + c_{i-1}]\!]$, where c_{i-1} is the carry value. We have that $z_i = t \bmod 2$, and $c_i = \lfloor t/2 \rfloor$.

$$z_i = x_i + y_i + c_{i-1} - 2x_iy_i - 2x_ic_{i-1} - 2y_ic_{i-1} + 4x_iy_ic_{i-1}$$

$$c_i = x_iy_i + x_ic_{i-1} + y_ic_{i-1} - 2x_iy_ic_{i-1}$$

If both x and y are private, all of these terms can be computed using the "simple" multiplication protocol. We need 4 such multiplications for each bit. So, $O(n)$ in total, using n rounds. If only one is private, then we need one dichotomous multiplication. If both are shared, then we use the dichotomous multiplication all the time.

Similarly, multiplication of two numbers x,y is achieved by the school method. This requires $O(n^2)$ bit multiplications. We omit further details.

10.4 Inverse

Given an encrypted value $[\![x]\!]$ we wish to compute $[\![1/x]\!]$, assuming that $x \neq 0$.

Known solution (see [CDN01]): generate jointly random $[\![r]\!]$. Use strong multiplication to compute $[\![rx]\!]$ and decrypt to get rx. If $rx = 0$ then abort, otherwise compute $[\![1/(rx)]\!]$. Finally, use strong multiplication with $[\![r]\!]$ to obtain $[\![1/x]\!]$.

Using $p(x) = x^{q-2}$ as reduction polynomial, we get an alternative solution requiring $\log q$ strong multiplications. This solution also works for $x = 0$, yielding $[\![0]\!]$ as result.

10.5 Zero test

Given an encrypted value $[\![x]\!]$ we wish to test whether $x = 0$ or not. The result should be encrypted as well, so the result is $[\![[x = 0]]\!]$. ($[x = 0] = \mathrm{sgn}(x)^2$)

The reduction polynomial is $p(x) = x^{q-1}$. Efficiency is $O(\log q)$ strong multiplications. Let $\delta(x) = x^{q-1}$.

If the result need not be encrypted, then use socialist millionaires.

Note that our solution for socialist millionaires with encrypted output assumes that the input values are given bit-by-bit. Efficiency is $O(n)$ dichotomous multiplications.

10.6 Parity bit

Given an encrypted value $[\![x]\!]$ we wish to determine the parity of x (when viewed as an integer in $\{0, \ldots, q-1\}$). We also show how to get an encrypted answer.

If x is private to a single player, the player proves in zero-knowledge that it revealed the parity of x correctly. Write $x = 2x' + x_0$, and create commitments for x' and x_0 that can be

verified for correctness w.r.t. $[\![x]\!]$. Prove that $x_0 \in \{0,1\}$. Prove that $x' \in \{0,\ldots,(q-1)/2\}$. The latter proof can be done using an efficient interval proof.

If x is shared and sufficiently small ($x < \sqrt{q}$), we have the following protocol. Jointly generate an even random number $[\![r]\!]$, $0 \leq r < q/2$ (player P_i generates a random number $[\![r_i]\!]$, $0 \leq r_i < q/8$, plus a proof; then take $[\![2(r_1 + r_2)]\!]$, which is an even number by construction.) Then decrypt $[\![x+r]\!]$ to learn the parity of x. Since r is from a much larger range than x, the value of $x+r$ is statistically independent of x (only negligibly dependent).

The above method is extended for an encrypted answer as follows. Player P_i generates a random $\{0,1\}$ value $[\![s_i]\!]$ plus proof. Instead of decrypting $[\![x+r]\!]$, we now decrypt $[\![x+r+s_1+s_2]\!]$, and encrypt the parity of $x+r+s_1+s_2$ as $[\![\mathrm{lsb}(x+s_1+s_2)]\!]$. Next, we use the xor-homorphic protocol for $[\![s_1]\!]$ and $[\![\mathrm{lsb}(x) \oplus s_1 \oplus s_2)]\!]$, then for $[\![s_2]\!]$ and $[\![\mathrm{lsb}(x) \oplus s_2)]\!]$ to get $[\![\mathrm{lsb}(x)]\!]$.

Note that the parity of x can also be determined by computing $[\![x/2]\!]$ and then testing whether the encrypted value is below (even) or above (odd) $q/2$, when viewed as an integer in $\{0,\ldots,q-1\}$.

The reduction polynomial follows from $p(x) = \delta(\prod_{i=0}^{(q-1)/2}(x-2i))$.

11 Advanced operations

11.1 Hamming distance

Given two vectors x and y with entries in $\mathbb{Z}_q$, the Hamming distance $d_H(x,y)$ between x and y is defined as follows

$$d_H(x,y) = \sum_{i=1}^{n} \delta(x_i - y_i),$$

where $\delta(x) = 0$ if $x = 0$ and $\delta(x) = 1$ if $x \neq 0$. The goal of this section, is to compute securely the Hamming distance between x and y without revealing any further information about x and y. More precisely, we assume that there are two players P_1 and P_2 each having a vector, say x and y respectively. They want to compute $d_H(x,y)$ by performing a two-party protocol. We denote the i-th entry of the vector x by $x_i \in \mathbb{Z}_q$. The entries x_i can be represented as binary strings through the following representation:

$$x_i = \sum_{j=0}^{m} x_{ij} 2^j.$$

The equality tests are done as in Section ??, using variable h.

At the end of the protocol P_1 and P_2 decrypt h. Invariant:

$$h = \delta(\sum_{k=0}^{j}(x_k - y_k)^2).$$

In order to compute $d_H(x,y)$, the above defined protocol has to be performed for every entry x_i, y_i of the vectors x and y. Denote the outcomes for the entry i by h_i. Then,we have

$$[\![d_H(x,y)]\!] = [\![\sum_i h_i]\!] = \prod_i [\![h_i]\!].$$

Then both players decrypt together $[\![d_H(x,y)]\!]$.

Details:

In this section, we investigate the problem of computing securely the Hamming distance between two vectors $x, y \in \mathbb{Z}_q^n$. The difference with the previous setting is that one of the players (say player) should after the protocol only know whether $d_H(x,y) \geq \mu$ where μ is a threshold choosen by him.

P_1 and P_2 follow therefore the following steps:

1. P_1 and P_2 compute securely following the protocol given in section ?? the data $[\![h_i]\!] = [\![\delta(x_i - y_i)]\!]$.
2. P_1 and P_2 compute securely the bit representation of the sum $[\![h_i]\!]$. Therefore they write $[\![h_i]\!]$ as $([\![0]\!], ..., [\![h_i]\!])$. Then, they compute the XOR sum of the bit strings $([\![0]\!], ..., [\![h_i]\!])$ for $i = 1, ..., n$. This can be done using the binary arithmetic protocols of the previous paragraph. Denoting the sum $\sum_i h_i$ by s, this protocol ends with the encrypted bit representation of the sum s, i.e. $([\![s_n]\!], ..., [\![s_0]\!])$.
3. P_2 chooses his threshold $\mu \in \mathbb{Z}_q$, computes the bit representation of μ and encrypts it, $([\![\mu_n]\!], ..., [\![\mu_0]\!])$.
4. P_1 and P_2 carry out the "Millionaires" protocol based on multivariate polynomials as developed above. Then, they decrypt the result of this protocol.

11.2 Euclidean Distance

In this section, we consider the same problem as in Section 11.1 only the Hamming distance is replaced by the Euclidean distance. We recall that for two vectors of length n over $\mathbb{Z}_q$, the Euclidean distance $d_E(x,y)$ is defined as

$$d_E(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2.$$

The players perform the following steps:

1. P_1 computes $[\![x_i^2]\!]$ from its knowledge of x_i and $[\![x_i]\!]$.
2. P_2 computes $[\![y_i^2]\!]$ from its knowledge of y_i and $[\![y_i]\!]$.
3. Compute $[\![h_i]\!] = [\![x_i^2 - 2x_iy_i + y_i^2]\!]$.

Finally, all the encryptions are multiplied and one obtains,

$$d_E(x,y) = \prod_{i=1}^{n} [\![h_i]\!] = [\![\sum_{i=1}^{n} h_i]\!]$$

By decrypting the result is obtained.

For shared inputs we need strong multiplication, or use bit-by-bit representations of x_i and y_i.

11.3 Other

Maximum and Sorting; intersection of sets, membership tests are also possible using the described invention.

12 Applications

12.1 Secure Profile Matching

In recent years, the availability to users of large amounts of content (audio, video, text, etc) in electronic form has called for the development of methods for information selection. Such methods are most commonly based on the idea of personalisation, where information is selected for a given user according to the profile of preferences of that user. Such systems are generally known as recommender systems (described by Furner).

Collaborative filtering (described by Griffith) techniques are recommender systems in which the recommendation of content is based on the similarity between the profile of a given user and the profiles of other users (and not in the features of the content itself). If the measure of similarity between any two profiles is high enough (according to some predefined criterion), the system can recommend to one user the highly appreciated content items of the other user, which have not yet been seen by that first user.

Here we extend this setting to the ad-hoc case where two users can compare their profiles and find out whether they have a similar taste. If so, they might start a procedure to exchange content with eachother. If not, the protocol guarantees that no other private information is leaked than the fact that the profiles are not similar.

In this section, we will give a few similarity measures for profile matching and show how the private matching problem can be solved using the techniques developped in the previous sections.

12.2 Similarity Measures

In this section, we show how to apply our techniques to the case of two users who want to compare their profiles in fully private way. From an application point of view, the main focus point of our approach is to allow for ad-hoc use. By private comparison of two profiles, we mean that the users compute securely a beforehand agreed testfunction. In a second phase they compare this (encrypted) value securely with a threshold; i.e. at the end of the protocol, the only knowledge the players get is whether the value of the testfunction exceeds the threshold or not. The participants are assumed to have an authenticated channel with each other. For sake of clarity we will restrict ourselves to the case where the private profiles of the users consist of binary vectors denoted as x and y but extensions to non-binary vectors are also possible.

A first measure for comparing two vectors is given by the number of entries in which they differ. This measure can be defined in terms of the Hamming distance $\Delta_1(x, y)$ between

two vectors $x, y \in \{0,1\}^n$ which is given by,

$$\Delta_1(x,y) = \sum_{i=1}^{n} \delta(x_i - y_i), \tag{5}$$

where $\delta(x) = 0$ if $x = 0$ and $\delta(x) = 1$ if $x \neq 0$. The second measure that we will consider in this paper is the scalar product defined as,

$$\Delta_2(x,y) = \sum_{i=1}^{n} x_i y_i. \tag{6}$$

The goal of this section is to show how Δ_1 and Δ_2 can be computed and compared to a threshold in a private way. The private computation of Δ_1 can be performed by running the private multiplier multiplication protcol 7.1 and using threshold decryption to decrypt the result. The private computation of Δ_2 is also based on the private multiplier multiplication protocol of section 7.1 and the homomorphic properties of the ElGamal crypto system.

A more interesting situation arises when the decision problem $\Delta_i(x,y) > \mu$ for $i = 1, 2$ and μ a threshold choosen by one or both of the players has to be solved in a private way. We assume that μ is given in its binary representation $(\mu_{n-1}, \ldots, \mu_0)$. We present a protocol that solves the decision problem for Δ_1, the situation for Δ_2 is completely analogous and the details are therefore omitted. The protocol looks as follows.

1. First, the players set up a threshold ElGamal system using the key generation protocol described in section ??.
2. For each component $i = 1, \ldots, n$ both players compute securely $[\![o_i]\!] = [\![\delta(x_i - y_i)]\!]$ using the "socialist" protocol of section 10.1 or by compute $[\![o_i]\!] = [\![(x_i - y_i)^2]\!]$ using the private multiplier multiplication protocol of section 7.1.
3. They compute privately the bit-representation of $[\![s]\!] = [\![\sum_{i=1}^{n} o_i]\!]$ by applying the arithmetic protocols of section ?? to $[\![o_i]\!]$. As a result the players obtain $([\![s_{n-1}]\!], \ldots, [\![s_0]\!])_2$; the binary representation of $[\![\sum_{i=1}^{n} o_i]\!]$.
4. The players carry out the millionaires protocol of section 10.2 on $[\![s]\!]$ and μ to check whether $[\![s]\!] \geq \mu$.
5. Finally, they apply (fair) threshold decryption to decrypt the result of the decision problem.

This protocol requires $\mathcal{O}(n \log n)$ exponentiations per player.

We note that the previous approach can be extendend to the case where the entries belong to a dicrete (bit not binary) domain. The idea is the same but the computations require more steps and details. We emphasize that also in that case, we can guarantee full privacy.

12.3 Auctions

In order to illustrate the wide applicability of our approach, we investigate a second application: secure auctions. We will mainly compare our results with those of [?] as they

presented the most efficient auction protocol this far. Their approach is based on the Mix and Match technique introduced by Juels and Jakobsson in [JJ00]. We show that our approach belongs to the fastest known (thus far) and is the fastest in computational sense. By using our multiplication protocols we avoid the relatively computationally expensive Mix computation. Our protocol satisfies the same advantages as formulated by Juels and Jakobsson [JJ00], in particular it satisfies: non interactivity, auction adaptiblity, full privacy, robustness, multiple servers and public verifiability.

An auction consists of two phases: a bidding phase during which the participants send their bids to the auctioneer, and an opening phase during which the auctioneer announces the highest price and the identity of the winner. We assume the following model. There are m bidders, $P_1, \ldots, P_m$. The bids are given by $x_1 = (x_{1,n-1}, \ldots, x_{1,0})_2, \ldots, x_m = (x_{m,n-1}, \ldots, x_{m,0})_2$. Note that the representations are ordered from msb to lsb. The bidders encrypt their bids with the joint public key of the servers, and send those to the auctioneer; $[\![x_i]\!] = ([\![x_{i,n-1}]\!], \ldots, [\![x_{i,1}]\!])$. There are k servers. We will describe our method for highest price auctions and discuss briefly how it can be extended to Vickrey (second-price) auctions.

We construct an algorithm for determining the identity of the highest bidder. This algorithm is used by the servers to determine securely the highest bid and the identity of the highest bidder(s). Therefore, we define a set of $n+1$ selection vectors $w_i \in \{0,1\}^m$, $i = -1, \ldots, n-1$ that keep track of the identities of the highest bidder up to bit i (starting from the msb). The algorithm starts with the vector w_{n-1} and the identity of the highest bidder is contained in the vector w_{-1}. In order to give the dynamics that updates w_i to w_{i-1}, we define a second set of vectors $t_i \in \{0,1\}^{m+1}$ $i = 0, \ldots, n-1$. The vectors t_i check whether the vector $x_j w_j$ equals the zero vector. We denote the j-th component of the vectors w_i, t_i by $w_{j,i}, t_{j,i}$. The initial condition for the t vectors is given by, $t_{0,j} = 0$ for $j = 0, \ldots, n-1$ and for the w-vectors is given by $w_{n-1} = (1, \ldots, 1)$. We define the polynomials $F(s,z) = s + (1-s)z$ and $G_a(s,z) = s(z + (1-z)(1-a))$. The dynamics is then defined by the following updating rule,

$$t_{j,i} = F(t_{j-1,i}, x_{j,i} w_{j,i}), \quad t_{m,i} = F(\ldots F(F(t_{0i}, x_{1i} w_{1i}), x_{2i} w_{2i}) \ldots),$$
$$w_{j,i-1} = G_{t_{m,i}}(w_{j,i}, x_{j,i}),$$

for $i = n-1, \ldots, 0$ starting with $i = n-1$ and for each i, the counter j runs from 1 to m. Note that $t_{mi} = 1$ means that at least one of the components of the vector $x_j w_j$ equals one. In order to compute the vector w_{-1} securely, the servers use the generalized millionaires protocol of section ?? based on the conditional gate. When the vector w_{-1} has been computed securely, the servers use fair threshold decryption to decrypt the entries of the vector w_{-1}. The identities of the winning bidders correspond to the positions of the entries of w_{-1} that are equal to one. Using this identifier, they can find the corresponding highest bid and use threshold decryption to decrypt it. In order to evaluate the performance of our protocol we start by omitting the verification computations. The computations require $3mn$ dichotomous shared multiplier multiplication protocols; i.e. $51mn$ exponentiations per server. The fastest auction protocol up to now is based on the Mix and Match approach and described in [?]. Their protocol requires mn AND gates. Adding all

exponentiations for a secure evaluation of the AND gate, consisting of the exponentiations for threshold decryption, four PETs (Plaintext Equality Test), a MIX protocol together with the necessary Zero-Knowledge proofs, amounts to 151 mn exponentiations. Hence, the protocol proposed in this paper is three times as fast as the protocol given in [?], which on its turn is seven times faster than the Mix and Match approach of [JJ00]. Taking the verification efforts into account an extra factor k has to be added. Finally, we mention that the total number of rounds for the servers computations is given by $3mnk$.

Now, we return to Vickrey auctions. We remind the reader that a Vickrey auction is an auction where the highest bidder wins but the clearing price, i.e. the price that the winner has to pay, is equal to the second highest bid. In order to perform a Vickrey auction, one can follow the following approach. First the servers determine the identities of the winners (but not the winning bids) with the protocol given above. Then, they remove the winners and their bids from the list. Finally, they evaluate the following set of polynomials,

$$p_j = F\left(....F(F(0, x_{1,j}w_{1,j}), x_{m,j}w_{m,j}\right),$$

for $j = n-1, \ldots, 0$ and where F is as defined above. The vector $p = (p_{n-1}, \ldots, p_0)$ contains then the maximum bid price. Note that this procedure adds only mn conditional gates, i.e. $17mn$ exponentiations. The approach of [?] yields $2mn$ AND gates which gives $300\,mn$ exponentiations.

12.4 Applications: Membership tests

We will very brief about this topic. The main question is to check securely whether an encrypted element $[\![x]\!]$ belongs to a certain set A. The set A consists of encrypted values only.

By applying the techniques earlier mentioned in this document, this problem can be tackled too.

13 General (Strong) Multiplication

Given two homomorphic encryptions $[\![x]\!], [\![y]\!]$ the object is to compute a homomorphic encryption $[\![xy]\!]$. We use a protocol from [CDN01].

1. Player P_i chooses a random value r_i and sends $[\![r_i]\!]$ to player P_{3-i} along with a proof that it knows r_i, for $i = 1, 2$.
2. The players jointly decrypt $[\![x + r_1 + r_2]\!]$.
3. Let $x_1 = x + r_2$, $x_2 = -r_2$. Player P_i sends $[\![x_i]\!]$, $[\![f_i]\!] = x_i[\![b]\!]$ to player P_{3-i} along with a proof, for $i = 1, 2$.
4. Both players may compute $[\![f_1]\!] + [\![f_2]\!] = [\![xy]\!]$.

If any of the proofs fails, the protocol is aborted.

The protocol takes a constant number of modular exponentiations. Here's an exact count in case Paillier's cryptosystem is used.

1. $[\![r_i]\!] = g^{r_i} y^n \bmod n^2$, for random y.
2. Proof of knowledge of r_i: another one.
3. Verification of proof: another one.
4. Decrypt per player: one exponentiation.

Euclidean Distance Here we discuss the same problem as in the previous subsection. The only difference consists in the fact that instead of the Hamming distance we use the Euclidean distance as a measure of similarity. We recall that for two vectors x and y of length n, the Euclidean distance $d_E(x, y)$ is defined as,

$$d_E(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2. \tag{7}$$

In order to compute $d_E(x, y)$ the players perform the following steps:

1. Player 1 computes $[\![x_i^2]\!]$ for all $i = 1, \ldots, n$ from its knowledge of the x_i. Similarly player 2 computes $[\![y_i^2]\!]$ for all $i = 1, \ldots, n$.
2. Both players compute $[\![o_i]\!] = [\![x_i^2 - 2x_i y_i + y_i^2]\!]$. Therefore player 1 sends $[\![x_i]\!]$ to player 2 who can then compute $[\![2x_i y_i]\!]$ (together with a proof that she used the correct y_i, i.e. the same one as she used in the computation of y_i^2.) Then they compute $[\![x_i^2 - 2x_i y_i + y_i^2]\!]$.
3. Finally, they compute $[\![d_E(x, y)]\!]$ by making use of the homomorphic properties of the encryption scheme as follows,

$$[\![d_E(x, y)]\!] = \prod_{i=1}^{n} [\![o_i]\!] = [\![\sum_{i=1}^{n} o_i]\!]. \tag{8}$$

4. By using fair threshold decryption, the result is obtained.

Again, we consider also the threshold version of this computation, i.e. the case where one (or both) of the players only get the answer to the decision problem $d_E(x, y) > \mu$ for some threshold μ. In that situation, both players have to use the binary representation of their inputs and compute in binary representation the values of the outcomes $[\![o_i]\!]$. Then, they compute the binary representation of $\sum_{i=1}^{n} [\![o_i]\!]$ by using the same methodes as explained in the previous section. Then, they carry out the "Millionaires" protocol of section 10.1 to obtain the encrypted result. Finally, they use fair threshold decryption to reveal the solution to both players.

Scalar product Another well-known similarity measure for comparing two vectors is the (normalized scalar product) which is defined as follows,

$$\langle x, y \rangle = \frac{\sum_{i=1}^{n} x_i y_i}{||x||\,||y||} \tag{9}$$

where $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$. As the data x, y are private, the numbers $1/||x||$ and $1/||y||$ can be computed privately by the respective players. The sum $[\![\sum_{i=1}^{n} x_i y_i]\!]$ can be computed using

the homomorphic properties of the El Gamal encryption scheme. Using the homorphic properties once more, one obtains $[\![\langle x, y\rangle]\!]$. Finally, the value is obtained by applying (fair) threshold decryption and by extending the technique of [FSW01] to deal with rational numbers to the El Gamal case.

In order to solve the associated decision problem, i.e. to decide whether $\langle x, y\rangle > \mu$ for some well defined threshold μ all computations have to be done in the binary representation as explained before. Moreover as $0 \leq \mu \leq 1$, it looks favorable to solve the following associated decision problem: $\frac{1}{\mu}\sum_{i=1}^{n} x_i y_i \geq ||x||\,||y||$. Then the "Millionaires" protocol of section 10.1 has to be applied. Finally, the result is obtianed by applying (fair) threshold decryption.

References

[Bau01] O. Baudron and J. Stern, *Non-interactive Private Auctions*, Financial Cryptography 2001, vol. 2339 LNCS, 364-378, Berlin 2001, Springer-Verlag.

[BST01] F. Boudot, B. Schoenmakers, and J. Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1–2):23–36, 2001. Special issue on Coding and Cryptology.

[CD01] R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. In *Advances in Cryptology—CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 119–136, Berlin, 2001. Springer-Verlag.

[CDN01] R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300, Berlin, 2001. Springer-Verlag.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.

[DN03] I. Damgård and J.B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—CRYPTO '2003*, volume 2729 of *Lecture Notes in Computer Science*, pages ???–???, Berlin, 2003. Springer-Verlag.

[FH96] M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, 1996.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310, Berlin, 1999. Springer-Verlag.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[JJ00] A. Juels and M. Jakobsson. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology—ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177, Berlin, 2000. Springer-Verlag.

[KMO01] J. Katz, S. Myers, and R. Ostrovsky. Cryptographic counters and applications to electronic voting. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 78–92, Berlin, 2001. Springer-Verlag.

[Ped91] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1991. Springer-Verlag.

[Yao82] A. Yao, *Protocols for secure computations*, In Proc. 23rd IEEE Symposium of Computer Science (FOCS 82), 160-164, IEEE Computer Society, 1982.

[CGS97] R. Cramer, R. Gennaro, B. Schoenmakers, *A secure and efficient multi-authority election scheme*, In Advance in Cryptology Eurocrypt 97, vol. 1233 LNCS, 103-118, 1997, Springer-Verlag, Berlin.

[FSW01] P-A. Fouque, J. Stern, G-J. Wackers, *CryptoComputing with rationals*, Financial Cryptography, 2001.

[handbook] Handbook of applied cryptography, Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, CRC Press, ISBN: 0-8493-8523-7, October 1996, 816 pages, Fifth Printing (August 2001)

[Stin02] Doug Stinson, Cryptography Theory and Practice, Second Edition , published by CRC Press in March, 2002.
1.Turner
2.Griffith

It should be noted that the above-mentioned embodiments illustrate rather than limit the invention, and that those skilled in the art will be able to design many alternative embodiments without departing from the scope of the appended claims.

In the claims, any reference signs placed between parentheses shall not be construed as limiting the claim. The word "comprising" does not exclude the presence of elements or steps other than those listed in a claim. The word "a" or "an" preceding an element does not exclude the presence of a plurality of such elements. The invention can be implemented by means of hardware comprising several distinct elements, and by means of a suitably programmed computer. A single processor or other (programmable) unit may also fulfill the functions of several means recited in the claims.

In the device claim enumerating several means, several of these means can be embodied by one and the same item of hardware. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage.

24

Claims

1. Method for multiparty computation, substantially as described above.
2. Method for allowing at least two users to compare their private data without revealing any other information than whether they are similar or not, according to some measure.
3. Method for at least two users to obtain the product of two possibly encrypted numbers and optionally a proof that is was correctly computed and optionally that the other player did not cheat.
4. Device to support or implement any of the methods of claims 1-3.
5. System to support or implement any of the methods of claim 1-3.
6. Signal carrying protocol information related to the protocols of claim 1-3.